Machine Learning System Design

전 재 욱

Embedded System 연구실
성균관대학교

Embedded System Lab.

Outline

- Error Analysis
- Error metrics for skewed classes
- Data for machine learning

Outline

- Error Analysis
- Error metrics for skewed classes
- Data for machine learning

Recommended Approach

- Start with a simple algorithm that we can imp quickly.
  - Implement it and test it on our cross-validation data.

- Plot learning curves
  to decide if more data, more features, etc. are likely to help.

- Error analysis
  - Manually examine the examples (in cross validation set) that our algorithm made errors on.
  - See if we spot any systematic trend in what type of examples it is making errors on.

Outline

- Error Analysis
- Error metrics for skewed classes
- Data for machine learning

Machine learning system design

- **Error metrics for skewed classes**

Error Metrics for Skewed Analysis

- Example of skewed classes
  - (Number of examples in one class)

    << (Number of examples in the other)

    ➔ So standard error metrics are not so good

Cancer Classification Example

- Train logistic regression model $h_\theta(x)$
  - $(y = 1$ if cancer, $y = 0$ otherwise)
- Find that you got 1% error on test set.
  - (99% correct diagnoses)
    - This looks pretty good.
- Only 0.5% of patients have cancer

```
function y = predictCancer(x)
        y = 0; %ignore x!
return
```

→ 0.5% error

  - Now, 1% error looks very bad.
- When the number of examples in one class is very small
  - this is an example of skewed classes

Another Example

- Algorithm has 99.2% accuracy
  - Make a change, now get 99.5% accuracy
    - Does this really represent an improvement to the algorithm?

- Did we do something useful,

  or did we just create something which predicts y = 0

  more often
  - Get very low error, but classifier is still not great

Precision and Recall

- $y = 1$ in presence of rare class that we want to detect

		Actual Class	
		1	0
Predicted Class	1	True Positive	False Positive
	0	False Negative	True Negative

- Precision
  - Of all patients where we predicted $y = 1$, what fraction actually has cancer?
  - $\frac{True\ positives}{\#\ of\ predicted\ positive} = \frac{True\ positive}{True\ pos + False\ \mathbf{Pos}}$
- Recall
  - Of all patients that actually have cancer, what fraction did we correctly detect as having cancer?
  - $\frac{True\ positives}{\#\ of\ actual\ positive} = \frac{True\ positive}{True\ pos + False\ \mathbf{neg}}$

Precision and Recall

- Precision
  - *How often does our algorithm cause a false alarm?*

  - Of all patients we predicted have cancer,
    what fraction of them *actually* have cancer?
    - $\frac{True\ positives}{\#\ of\ predicted\ positive} = \frac{True\ positive}{True\ pos + False\ \boldsymbol{Pos}}$

  - High precision is good (i.e. closer to 1)
    - We want a big number
      - because we want $False$ $\boldsymbol{Positive}$ to be as close to 0 as possible

Precision and Recall

- Recall
  - *How sensitive is our algorithm?*
  - Of all patients in set that actually have cancer, what fraction did we correctly detect
    - $\frac{True\ positives}{\#\ of\ actual\ positive} = \frac{True\ positive}{True\ pos + False\ \mathbf{neg}}$
  - High recall is good (i.e. closer to 1)
    - We want a big number
      - because we want $False\ \boldsymbol{negative}$ to be as close to 0 as possible

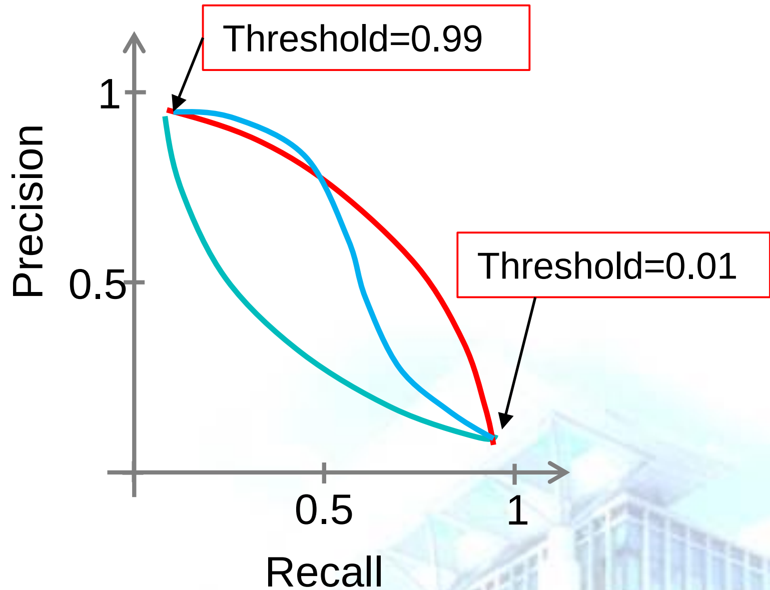
Precision and Recall

- By computing precision and recall
  get a better sense of how an algorithm is doing
  - Means we are much more sure that an algorithm is good
- Typically, the presence of a rare class
  is what we are trying to determine
  - e.g. positive (1) is the existence of the rare thing

Trading off Precision and Recall

- Logistic regression: $0 \leq h_\theta(x) \leq 1$
  - Predict 1 if $h_\theta(x) \geq 0.5$
  - Predict 0 if $h_\theta(x) < 0.5$
- Suppose we want to predict $y = 1$ (cancer) only if very confident.
  - Predict 1 if $h_\theta(x) \geq 0.8$
  - Higher precision, lower recall
    - Risk of false negatives
- Suppose we want to avoid missing too many cases of cancer (avoid false negatives).
  - Predict 1 if $h_\theta(x) \geq 0.3$
  - Higher recall, lower precision
    - Risk of false positives
- More generally:
  - Predict 1 if $h_\theta(x) \geq thresholed$

- Precision = $\frac{True\ positives}{\#\ of\ predicted\ positive}$
- Recall = $\frac{True\ positives}{\#\ of\ actual\ positive}$



Curve shape can be changed depending on classifier details

Trading off Precision and Recall

- How to compare precision/recall numbers?
  - Which algorithm is the best among the following three?

	Precision(P)	Recall(R)
Algo 1	0.5	0.4
Algo 2	0.7	0.1
Algo 3	0.02	1.0

Trading off Precision and Recall

- How to compare precision/recall numbers?
  - Average: $\frac{P+R}{2}$
    - 0.45, 0.4, 0.51
      - 0.51 is the best despite having a recall of 1 - i.e. predict y=1 for everything
    - NOT good

	Precision(P)	Recall(R)	Average
Algo 1	0.5	0.4	0.45
Algo 2	0.7	0.1	0.4
Algo 3	0.02	1.0	0.51

Trading off Precision and Recall

- How to compare precision/recall numbers?
  - F_1 Score (F score): $2\frac{PR}{P+R}$ $\left(\leftarrow \frac{2}{\frac{1}{p}+\frac{1}{R}}\right)$
    - F_1 score is like taking the average of precision and recall giving a higher weight to the lower value
    - P=0 or R=0
      ➔ F_1 score = 0
    - P=1 **and** R=1
      ➔ F_1 score = 1

	Precision(P)	Recall(R)	F Score
Algo 1	0.5	0.4	0.2222
Algo 2	0.7	0.1	0.0875
Algo 3	0.02	1.0	0.0196

Trading off Precision and Recall

- How to compare precision/recall numbers?
  - F_1 Score (F score): $2\frac{PR}{P+R}$ $\left(\leftarrow \frac{2}{\frac{1}{p}+\frac{1}{R}}\right)$
- If we are trying to automatically set the threshold, one way is to try a range of threshold values and evaluate them on our cross validation set
  - Then pick the threshold which gives the best F_1 score.

Outline

- Error Analysis
- Error metrics for skewed classes
- Data for machine learning

Designing A High Accuracy Learning System

- Classify btw confusable words
  - {to, two, too}, {then, than}
  - For breakfast I ate _____ eggs.

- Algorithms
  - Perceptron (logistic regression)
  - Winnow
    - Like logistic regression
    - Used less now
  - Memory based
    - Used less now
  - Naive Bayes

[Banko and Brill, 2001]

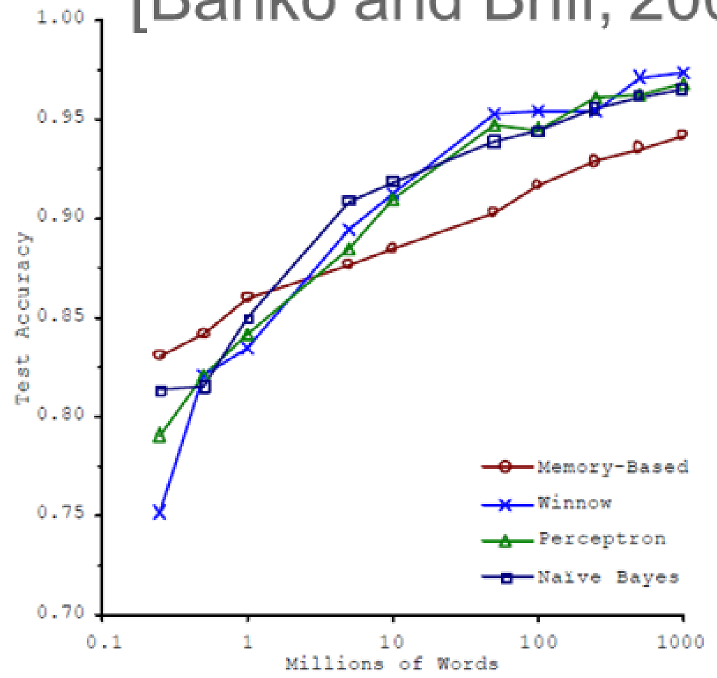


Figure 1. Learning Curves for Confusion Set Disambiguation

Designing A High Accuracy Learning System

- What can we conclude
  - Algorithms give remarkably similar performance
  - As training set sizes increases, accuracy increases
  - Take an algorithm, give it more data, should beat a "better" one with less data
  - Shows that
    - Algorithm choice is pretty similar
    - More data helps

[Banko and Brill, 2001]

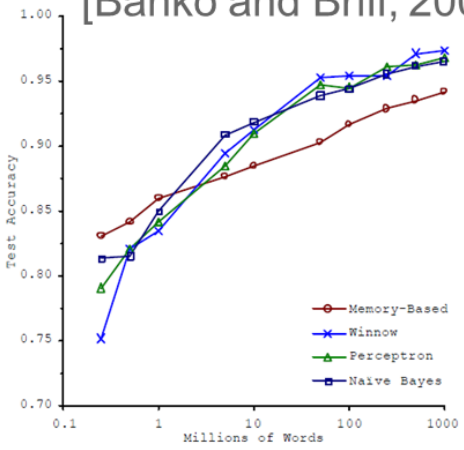


Figure 1. Learning Curves for Confusion Set Disambiguation

➔ It's not who has the best algorithm that wins.

It's who has the most data.

Large Data Rationale

- Assume feature $x \in R^{n+1}$ has sufficient information to predict y accurately.
  - Then more data may help
    - Example:
      - For breakfast I ate _____ eggs.
    - Counterexample:
      - Predict housing price from only size ($feet^2$) and no other features.

- Useful test:
  - Given the input x, can a human expert confidently predict y ?

Large Data Rationale

- Use a learning algorithm with ***<u>many parameters</u>***
  - e.g. logistic regression/linear regression with many features
  - neural network with many hidden units

  - A powerful learning algorithm with many parameters which can fit complex functions
    - Low bias algorithm
      - Little systemic bias in their description – flexible
      - $J_{train}(\theta)$ will be small

  - Use a very large training set
    - Unlikely to overfit
      - $J_{train}(\theta) \approx J_{test}(\theta)$

$J_{test}(\theta)$ will be small

Large Data Rationale

- An algorithm with having both low bias and low variance
  - Use complex algorithm
    - For low bias
  - Use large training set
    - For low variance

References

- Andrew Ng, https://www.coursera.org/learn/machine-learning
- http://www.holehouse.org/mlclass/11_Machine_Learning_System_Design.html
- M. Banko and E. Brill, “Scaling to Very Very Large Corpora for Natural Language Disambiguation,” 2001.